TURBO²

นิพันธ์ วงศ์ทองศิริ

ในยุทธจักรโปรแกรมคอมพายเลอร์ษาปาสคาล บนเครื่องระดับไมโครฯ เทอร์–ปาสคาลดูจะเป็นพระเอกยอดนิยมในขณะนี้ ายท่านยอมรับการใช้งานที่ง่าย รวดเร็ว ใจ แต่พระเอกของเราก็ไม่เอาใจแฟนไปทุกอย่าง โดยเฉพาะในส่วนของการจัดเกี่ยวกับจอภาพ เทอร์โบมีกระบวนความจัดจอภาพให้ใช้งานแค่ 9 กระบวนความนั้น ซึ่งผมคิดว่าน้อยเกินไป

กระบวนความต่างๆ ที่เขียนขึ้นจากความนี้จึงเป็นกระบวนความเพิ่มเติม ทำการควบคุมการแสดงผลบนจอภาพง่ายะมีประสิทธิภาพดียิ่งขึ้น เช่น กระบวนมที่ทำการลบจอภาพแค่ 24 บรรทัด โดยรทัดที่ 25 จะเก็บเอาไว้แสดงหน้าที่ของก์ชันคีย์ หรือกระบวนความที่ทำให้ข้อมหรือตัวอักษรบนจอภาพกะพริบ นอกือจากการแสดงแบบอักษรแสงเข้มหรือจ่อนที่เทอร์โบมีอยู่แล้ว และอีกหลายะบวนความที่คุณผู้อ่านจะได้ประโยชน์บบทความต่อไปนี้

อของเครื่องไมโครฯ

ใจจริงผมก็อยากจะให้บทความนี้ใช้กับไมโครฯทุกยี่ห้อละครับ แต่ก็เป็นไปได้สำหรับผม ฉะนั้นบทความนี้จึงต้องนขึ้นสำหรับเครื่องไอบีเอ็ม พีซี เอ็กซ์ที หรือเครื่องยี่ห้อสารพัดที่เป็นไอบีเอ็มแพด 100% เพราะกระบวนความที่เขียนเขียนด้วยภาษาแอสเซมบลี ซึ่งมีการเรียกโปรแกรมย่อยใน BIOS(Basic Input/tput System) โดยตรง

เหตุที่เขียนกระบวนความ (Procedure)

เรื่องของเรื่องก็มาจาก ผมได้ซื้อเครื่องไอบีเอ็มคอมแพตมาใช้งานส่วนตัว ก่อนจะใช้งานผมก็ต้องทำความรู้จักกับเครื่อง จนไปถึงในส่วนของสกรีนไดรเวอร์(Screen Driver) ในคู่มืออธิบายการใช้งานของ ANSI.SYS ซึ่งเป็นไดรเวอร์สำหรับควบคุมจอภาพ แต่พอผมเอามาใช้กับเทอร์โบปาสคาลปรากฏว่าใช้ไม่ได้ผล ผมก็เลยเดาเอาว่า เทอร์โบคงจะจัดการเกี่ยวกับจอภาพโดยตรง โดยไม่ผ่าน ANSI.SYS (หากไม่เป็นตามนี้ต้องขออภัยด้วย เพราะผมเพียงคาดคะเนเอาเท่านั้น)

กระบวนความที่เขียนขึ้น ผมได้ทดลองใช้งานจริง นอกเหนือจากตัวอย่างของแต่ละกระบวนความ ซึ่งยังไม่พบปัญหาผิดพลาด สำหรับความเร็วและขนาดของโปรแกรมก็เป็นที่พอใจ คือมีทั้งความเร็วและขนาดกะทัดรัด หากท่านผู้อ่านที่ไม่คุ้นกับภาษาแอสเซมบลี (ซึ่งก็เหมือนผม) ก็สามารถเขียนด้วยเทอร์โบปาสคาลได้ เพียงแต่ว่าความเร็วและขนาดของโปรแกรม จะทำงานได้ช้ากว่าและมีขนาดใหญ่กว่าเท่านั้น (หลังจากที่ผ่านการแปลเป็นภาษาเครื่องโดยเทอร์–โบปาสคาล)

ขอบเขตของการอธิบาย

การอธิบายในแต่ละกระบวนความนั้น ผมจะอธิบายในส่วนของการใช้งาน ส่วนการอธิบายในเรื่องของโปรแกรมนั้น คงจะเป็นเรื่องยาว เพราะจะต้องกล่าวถึงฮาร์ดแวร์ของไอบีเอ็ม พีซี เอ็กซ์ที และในส่วนซอฟท์–แวร์ของซีพียูเบอร์ 8086/8088 ด้วย หากท่านผู้อ่านบางท่านมีความสนใจเพิ่มเติมก็ขอให้อ่านบทความของอาจารย์บุญเลิศ เอี่ยมทัศนาในไมโครฯฉบับที่ 21 ในบทความชื่อ เทอร์โบปาสคาลกับภาษาเครื่อง ซึ่งกล่าวถึงรายละเอียดในการผ่านค่าพารามิเตอร์ เข้าสู่กระบวนความที่เขียนด้วยภาษาแอสเซมบลี ประกอบบทความนี้ด้วย

โปรแกรมอื่นๆที่ต้องมี

นอกเหนือจากเทอร์โบปาสคาลแล้ว ท่านผู้อ่านจะต้องมีโปรแกรมอื่นๆ ด้วยคือ

1) โปรแกรม Editor ซึ่งอาจจะใช้ของเทอร์โบปาสคาล หรือใช้เวิร์ดสตาร์ในส่วนของ non document file แต่ถ้าท่านผู้อ่านมีโปรแกรมไซด์คิก ก็ใช้ในส่วนของ notepad ซึ่งผมคิดว่าจะสะดวกกว่า เพื่อสร้างซอร์สโปรแกรมให้ชื่อเป็นไฟล์ .ASM

2) โปรแกรม MACRO ASSEMBLER 8086/88 คือ คอมพายเลอร์สำหรับแปลซอร์สโปรแกรม ให้เป็นโปรแกรมภาษาเครื่อง ซึ่งหลังจากผ่านการแปลแล้ว จะได้ไฟล์ชนิด .OBJ

3) โปรแกรม LINK ที่มีอยู่ในแผ่น PC-DOS ซึ่งเมื่อผ่านขบวนการ LINK แล้วจะได้ไฟล์ชนิด .EXE

4) โปรแกรม EXE2BIN ที่มีอยู่ในแผ่น PC-DOS เช่นกัน โปรแกรมนี้จะทำการเปลี่ยนไฟล์ชนิด .EXE ให้เป็นไฟล์ชนิด .BIN แล้วเปลี่ยนชื่อเป็นไฟล์ .COM

เอาละครับเพื่อมิให้ยืดเยื้อต่อไปจะขอเริ่มเลยนะครับ

Procedure Clr24

กระบวนความนี้จะคล้ายกับกระบวนความ ClrScr ที่มีอยู่ในเทอร์โบ ผลที่ได้ที่แตกต่างกันก็เพียงแต่ว่ากระบวนความ ClrScr จะทำการลบจอภาพทั้งหมด แต่ Clr24 จะทำการลบจอภาพทั้งหมดยกเว้นบรรทัดที่ 25 จากนั้นเคอร์เซอร์จะไปอยู่ตำแหน่ง มุมบนซ้ายสุดของจอภาพ ซึ่งบรรทัดที่ 25 นี้อา จะเอาไว้แสดงข้อความ หรือแสดงหน้าท ของฟังก์ชันคีย์ ที่ต้องแสดงตลอดเวลาใ ขณะที่โปรแกรมที่เขียนขึ้นกำลังทำงานอ สำหรับตัวโปรแกรมและตัวอย่างการใช้งา จะแสดงในรูปที่ 1 และรูปที่ 2 ตามลำดับ

```
CODE    SEGMENT
        ASSUME  CS:CODE

CLR24   PROC    NEAR
        PUSH    BP
        MOV     BP,SP

        MOV     AX,0B800H       ; segment of video ram
        MOV     ES,AX           ; es is now pointed to video segment
        ; clear action
        MOV     DI,0            ; di is pointed to first location
        MOV     CX,1920         ; cx = 24*80 (number of columns to clear)
        MOV     AL,' '
        MOV     AH,07H
        CLD
        REP     STOSW
        ; place the cursor at upper left corner of video display
        MOV     DX,0
        MOV     AH,2
        MOV     BH,0
        INT     10H
        ;
        MOV     SP,BP
        POP     BP
        RET     0
CLR24   ENDP
CODE    ENDS
        END
```

รูปที่ 1 โปรแกรมสำหรับกระบวนความ Clr 2

```
Program Test_Clr24;
var
   y    : byte;
   ch   : char;

   Procedure Clr24; external 'CLR24.COM';

begin
   ClrScr;
   for y := 1 to 25 do
     begin
       GotoXY(1,y); write('Line #',y:3);
     end;
   GotoXY(15,5); write('Press any key to Testing');
   read(kbd,ch);
   Clr24;                    ( test here )
   repeat until keypressed;
end.
```

รูปที่ 2 ตัวอย่างการใช้งานกระบวนความ Clr 24

Procedure InsertChr

กระบวนความนี้ จะคล้ายกับกระบวนความ InsLine โดย InsLine จะทำการแทรกบรรทัดว่าง 1 บรรทัดตรงบรรทัดที่เคอร์เซอร์อยู่ในขณะนั้น พร้อมทั้งเลื่อนบรรทัดที่เคอร์เซอร์อยู่ และบรรทัดอื่นๆ ที่อยู่ใต้บรรทัดเคอร์เซอร์ลงไปข้างล่างของจอภาพ แต่สำหรับ InsertChr จะทำการเลื่อนข้อความที่นับจากตำแหน่งของเคอร์เซอร์ไปทางขวามือ 1 คอลัมน์ โดยตำแหน่งของเคอร์เซอร์จะไม่เปลี่ยนแปลง และตำแหน่งที่เคอร์เซอร์อยู่จะเป็น Blank Character ทั้งนี้การเลื่อนข้อความไปทางขวามือนับจากตำแหน่งข เคอร์เซอร์ จะไม่กระทบถึงข้อความในบรร อื่นๆ และข้อความที่เลื่อนไปทางขวามือ ไม่เลื่อนไปยังบรรทัดถัดไป โปรแกรมแ ตัวอย่างจะแสดงในรูปที่ 3 และรูปที่ 4 ต ลำดับ

หมายเหตุ ผลของการ RUN โปรแกรม รูปที่4 จะมีการเลื่อนของเคอร์เซ ด้วย ทั้งนี้เพราะเกิดจากคำ Write(ch)

รูปที่ 3

โปรแกรมสำหรับกระบวนความ Insert Chr

```
CODE        SEGMENT
            ASSUME  CS:CODE

INSERTCHR   PROC    NEAR
            PUSH    BP                  ; save enviroment
            MOV     BP,SP
            PUSH    DS

            ; read the current cursor position
            MOV     AH,3                ; mode 3 = read cursor position
            MOV     BH,0
            INT     10H                 ; return a result in dx
            CMP     DL,79               ; compare a column with right edge
            JE      SHORT FINISH        ; if column = 79 then goto finsih
            ; calculate a location of si & di
            MOV     AX,0
            MOV     CX,0
            MOV     BX,DX
            MOV     AL,BH
            INC     AL
            MOV     CL,80
            IMUL    CL
            MOV     CL,1
            SAL     AX,CL
            SUB     AX,2
            MOV     DI,AX
            DEC     AX
            DEC     AX
            MOV     SI,AX
            ; calculate a number of word to move
            MOV     AX,0
            MOV     AL,79
            SUB     AL,BL
            CBW
            MOV     CX,AX
            ; inserted action starts here
            MOV     AX,0B000H           ; monochome display
            MOV     ES,AX
            MOV     DS,AX
            STD
            REP     MOVSW
            ; write a blank at cursor position
            MOV     AL,' '
            MOV     AH,07H
            MOV     [DI],AX
            ;
            POP     DS                          ; release enviroment
 FINISH:    MOV     SP,BP
            POP     BP
            RET     0
INSERTCHR   ENDP
CODE        ENDS
            END
```

รูปที่ 4

ตัวอย่างการใช้งานกระบวนความ Insert Chr

```
Program Test_InsertChr;
var
    i   : byte;
    ch  : char;

    Procedure InsertChr;   external 'INSERT.COM';

begin
  ClrScr;
  GotoXY(1,5); write('This is a example program.');
  GotoXY(1,10); write('Press any key to testing / <E> = exit');
  GotoXY(12,5);
  repeat
    read(kbd,ch);  ( ch is a dummy character, not important character.)
    InsertChr; write(ch);          ( test here )
{
  NOTE :  The 'INSERTCHR' inserts a blank character and shifts all characters
          to the right side but it is not move cursor position.
          The 'WRITE' displays a CH character and then moves the cursor
          position to the right side for one position.
}
  until (ch = 'E') or (ch = 'e');
end.
```

Procedure
ClrEop

กระบวนความนี้จะทำการลบจอภาพโดยจะทำการลบจอภาพตั้งแต่ตำแหน่งของเคอร์เซอร์ในขณะนั้น ไปทางขวามือจนไปสิ้นสุดที่ตำแหน่งของคอลัมน์สุดท้ายของบรรทัดที่ 24 (ขอเน้นนะครับว่าบรรทัดที่ 24 มิใช่บรรทัดที่ 25) แต่ถ้าท่านผู้อ่านต้องการลบจอภาพจนถึงสิ้นสุดที่คอลัมน์สุดท้ายของบรรทัดที่ 25 ก็เพียงแต่แก้โปรแกรมเล็กน้อย โดยเปลี่ยนคำสั่งในประมาณบรรทัดที่ 27 (ที่ผมใช้ประมาณเพราะบางท่านอาจจะนับบรรทัดไม่ตรงกับผม ซึ่งไม่ว่าจะนับตรงหรือไม่ ถ้าใกล้เคียงก็จะต้องเห็นคำสั่งที่ต้องการแก้ไขแน่นอนครับ)

จากคำสั่ง **MOV AX, 1920**

แก้เป็น **MOV AX, 2000**

โปรแกรมและตัวอย่างจะแสดงในรูปที่ 5 และ 6 ตามลำดับ

```
CODE    SEGMENT
        ASSUME  CS:CODE

CLREOP  PROC    NEAR
        PUSH    BP
        MOV     BP,SP

        MOV     AX,0B000H       ; SEGMENT OF VIDEO RAM
        MOV     ES,AX           ; POINT TO VIDEO SEGMENT
; read current cursor
        MOV     AH,3            ;
        MOV     BH,0
        INT     10H
; CALCULATE LOCATION ON SCREEN
        MOV     AL,80
        MOV     CL,DH
        IMUL    CL
        PUSH    AX
        MOV     AL,DL
        CBW                     ; CONVERT AL BYTE TO  AX WORD
        POP     BX
        ADD     AX,BX
; CALCULATE NUMBER OF COLUMN TO CLEAR

        PUSH    AX
        PUSH    AX
        MOV     AX,1920
        POP     BX
        SUB     AX,BX
        MOV     CX,AX
; CALCULATE OFFSET ON VIDEO RAM

        POP     AX
        PUSH    CX
        MOV     CL,1
        SAL     AX,CL           ; MULTIPLY BY 2
;
        MOV     DI,AX
        MOV     AL,' '
        MOV     AH,07H
        POP     CX
        CLD
        REP     STOSW
;
        MOV     SP,BP
        POP     BP
        RET     0
CLREOP  ENDP
CODE    ENDS
        END
```

รูปที่ 5 โปรแกรมสำหรับกระบวนความ Clr Eop

```
Program Test_ClrEop;
var
    y       : byte;
    ch      : char;

  Procedure ClrEop; external 'CLREOP.COM'; { test this routine }

begin
  ClrScr;                       { standard identifier in turbo }
  for y := 1 to 25 do
    begin
      GotoXY(1,y); write(y:3,' This is a example program.');
    end;
  GotoXY(15,5);
  read(kbd,ch);                 { press any key to test }
  ClrEop;                       { test here }
  repeat until keypressed;
end.
```

รูปที่ 6 ตัวอย่างการใช้งาน กระบวนความ Clr Eop

Procedure
CursorType

กระบวนความนี้ เป็นการเปลี่ยนขนาดของเคอร์เซอร์ที่ท่านผู้อ่านเห็นบนจอภาพ โดยผมกำหนดไว้ 3 ขนาด คือ

ขนาด mode 0 มีรูปแบบของเคอร์เซอร์ดั้งเดิมเมื่อเปิดเครื่อง

ขนาด mode 1 มีรูปแบบที่มองไม่เห็น กล่าวคือ มองไม่เห็นเคอร์เซอร์นั่นเอง

ขนาด mode 2 มีรูปแบบที่ใหญ่ที่สุดเท่าที่สามารถกำหนดได้ กล่าวคือมีขนาดพอๆ กับตัวอักษรบนจอภาพ

เหตุผลที่ต้องการเปลี่ยนรูปแบบหรือขนาดของเคอร์เซอร์ เพราะบางครั้งรูปแบบดั้งเดิมมีขนาดเล็กเกินไป ทำให้สังเกตเห็นลำบาก หรือบางครั้งไม่ต้องการให้เห็นเคอร์เซอร์ในขณะที่จอภาพเปลี่ยนไป เช่น ภาพเคลื่อนไหวที่ต้องการแสดงเฉพาะภาพที่ต้องการเคลื่อนเท่านั้น

โปรแกรมและตัวอย่างจะแสดงในรูปที่ 7 และ 8 ตามลำดับ

```
CODE        SEGMENT
            ASSUME  CS:CODE

CURSORTYPE  PROC    NEAR
            PUSH    BP
            MOV     BP,SP

            MOV     AX,[BP+4]   ; [BP+4] = MODE
            CMP     AX,2
            JNE     SHORT MODE1             ; if mode <> 2 then goto mode1
            ; MODE 2 = box cursor type
            MOV     CX,000DH
            JMP     SHORT SET
   MODE1:   CMP     AX,1
            JNE     SHORT MODE0             ; if mode <> 1 then goto mode0
            ; MODE 1 = invisible cursor type
            MOV     CX,1000H
            JMP     SHORT SET
            ; MODE 0 = default cursor type
   MODE0:   MOV     CX,0B0CH
            ;
     SET:   MOV     AH,1
            INT     10H

            MOV     SP,BP
            POP     BP
            RET     2
CURSORTYPE  ENDP
CODE        ENDS
            END
```

รูปที่ 7 โปรแกรมสำหรับกระบวนความ Cursor Type

```
Program Test_CurorType;

  Procedure CursorType(mode:byte); external 'CURTYPE.COM';

begin
  ClrScr; CursorType(0);
  write('Cursor Type mode = 0  --> ');
  repeat until keypressed;
  writeln;
  CursorType(1);
  write('Cursor Type mode = 1  --> (invisible)');
  repeat until keypressed;
  writeln;
  CursorType(2);
  write('Cursor Type mode = 2  --> ');
  repeat until keypressed;
  CursorType(0)
end.
```

รูปที่ 8 ตัวอย่างการใช้งานกระบวนความ Cursor Type

Procedure

Reverse, UndLine, Blink, RevBlink

ภายในเทอร์โบปาสคาลมีกระบวนความอยู่ 2 กระบวนความที่กำหนดความเข้มตัวอักษรที่แสดงบนจอภาพ คือกระบวนความ NormVideo และกระบวนความ LowVideo แต่บางครั้งการแสดงผลบนจอภาพ ต้องการรูปแบบที่ต้องการเน้น หรือให้สังเกตข้อความง่ายขึ้น กระบวนความทั้ง 4 ที่เพิ่มเติมขึ้นจะมีลักษณะการใช้งานเหมือนกัน แต่ผลที่ออกมาบนจอภาพจะต่างกัน และมีสิ่งเพิ่มเติมที่ต้องคำนึงถึงในการใช้กระบวนความทั้ง 4 นี้ คือ ผมเขียนขึ้นใช้งานกับเทอร์โบปาสคาล เวอร์ชัน 3.01 บนเครื่องไอบีเอ็ม พีซี เอ็กซ์ที ซึ่งผมไม่แน่ใจว่า จะใช้งานได้กับเทอร์โบ เวอร์ชัน 2.0 หรือเวอร์ชัน 1.0 ได้หรือไม่ ทั้งนี้เพราะในแต่ละเวอร์ชันของเทอร์โบมีการเปลี่ยนแปลง ผมเคยตรวจสอบโปรแกรมภาษาเครื่องในกระบวนความเดียวกัน แต่ต่างเวอร์ชันปรากฏว่า โปรแกรมภาษาเครื่องออกมาต่างกัน สำหรับท่านผู้อ่านที่ใช้เวอร์ชันที่มิใช่ 3.01 ก็ขอให้ลองดูนะครับ

Procedure Reverse มีผลให้ข้อความที่แสดงบนจอภาพจะแสดงแบบ Reverse mode กล่าวคือ จะเป็นตัวอักษรสีดำบนพื้นขาว

Procedure UndLine มีผลให้ข้อความที่แสดงบนจอภาพ จะแสดงข้อความโดยมีขีดเส้นใต้ด้วย

Procedure Blink มีผลให้ข้อความที่แสดงบนจอภาพกะพริบ

Procedure RevBlink มีผลให้ข้อความที่แสดงบนจอภาพแสดงแบบ Reverse mode พร้อมทั้งกะพริบด้วย

โปรแกรมจะแสดงในรูปที่ 9, 10, 11, 12 สำหรับตัวอย่างของโปรแกรม จะเป็นตัวอย่างรวมที่แสดงอยู่ในรูปที่ 13

```
CODE        SEGMENT
            ASSUME  CS:CODE

REVERSE     PROC    NEAR
            PUSH    BP
            MOV     BP,SP

            PUSH    AX
            MOV     AL,70H
            MOV     DS:[0008],AL
            POP     AX

            MOV     SP,BP
            POP     BP
            RET     0
REVERSE     ENDP
CODE        ENDS
            END
```

รูปที่ 9 โปรแกรมสำหรับกระบวนความ Reverse

```
CODE        SEGMENT
            ASSUME  CS:CODE

UNDLINE     PROC    NEAR
            PUSH    BP
            MOV     BP,SP

            PUSH    AX
            MOV     AL,01H
            MOV     DS:[0008],AL
            POP     AX

            MOV     SP,BP
            POP     BP
            RET     0
UNDLINE     ENDP
CODE        ENDS
            END
```

รูปที่ 10 โปรแกรมสำหรับกระบวนความ Und Line

```
CODE        SEGMENT
            ASSUME  CS:CODE

BLINK       PROC    NEAR
            PUSH    BP
            MOV     BP,SP

            PUSH    AX
            MOV     AL,87H
            MOV     DS:[0008],AL
            POP     AX

            MOV     SP,BP
            POP     BP
            RET     0
BLINK       ENDP
CODE        ENDS
            END
```

รูปที่ 11 โปรแกรมสำหรับกระบวนความ Blink

```
CODE        SEGMENT
            ASSUME  CS:CODE

REVBLINK    PROC    NEAR
            PUSH    BP
            MOV     BP,SP

            PUSH    AX
            MOV     AL,0F0H
            MOV     DS:[0008],AL
            POP     AX

            MOV     SP,BP
            POP     BP
            RET     0
REVBLINK    ENDP
CODE        ENDS
            END
```

รูปที่ 12 โปรแกรมสำหรับกระบวนความ Rev Blink

```
Program Test_Display;

  Procedure Reverse;   external 'REVERSE.COM';
  Procedure UndLine;   external 'UNDLINE.COM';
  Procedure Blink;     external 'BLINK.COM';
  Procedure RevBlink;  external 'REVBLINK.COM';

begin
  ClrScr;
  NormVideo; writeln('Normal Video Display (high intensity)');
  LowVideo;  writeln('Low Video Display (low intensity)');
  Reverse;   writeln('Reverse Video Display');
  UndLine;   writeln('Underline Video Display');
  Blink;     writeln('Blink Video Display');
  RevBlink;  writeln('Reverse & Blink Video Display');
end.
```

รูปที่ 13 ตัวอย่างของการใช้งานกระบวนความจากรูปที่ 9 ถึง 12

Procedure
Print, PrintLn

กระบวนความทั้ง 2 ที่จะกล่าวถึงนี้มีลักษณะการทำงานคล้ายกับกระบวนความ Write และกระบวนความ writeln ตามลำดับ แต่กระบวนความ Print และ PrintLn นี้เวลาใช้งานจะต้องกำหนดแอตตริบิ้วของตัวอักษรหรือข้อความด้วย อีกทั้ง Print และ PrintLn จะทำการเขียนตัวอักษรหรือข้อความไปบนหน่วยความจำจอภาพ(Video RAM) โดยตรง การแสดงผลบนจอภาพจึงมีความเร็วกว่ากระบวนความ write และ writeln ข้อแตกต่างระหว่าง Print และ PrintLn คือกระบวนความ Print หลังจากที่เขียนข้อความบนจอภาพ ตำแหน่งของเคอร์เซอร์จะอยู่ท้ายสุดต่อจากข้อความ สำหรับกระบวนความ PrintLn หลังจากที่เขียนข้อความบนจอภาพ ตำแหน่งของเคอร์เซอร์จะไปอยู่คอลัมน์ที่ 1 ของบรรทัดถัดไป ทั้งกระบวนความ Print และ PrintLn นี้จะใช้เขียนข้อความได้ยาวสูงสุด 80 ตัวอักษร และบรรทัดที่สามารถเขียนได้สูงสุดคือบรรทัดที่ 24 หากการเขียนข้อความที่มีการเขียนเกินกว่าบรรทัดที่ 24 ก็จะมีการเลื่อนของจอภาพขึ้นไป 1 บรรทัด โปรแกรมทั้ง 2 กระบวนความจะแสดงในรูปที่ 14 และ 15 สำหรับตัวอย่างจะแสดงในรูปที่ 16

รูปที่ 16 ตัวอย่างการใช้งานกระบวนความ Print และ Print Ln

```
Program Test_Print_and_Println;
type
  string80      = string[80];
var
  sentense      : string80;
  ch            : char;

  Procedure Print(att:byte; text:string80); external 'PRT.COM';

  Procedure PrintLn(att:byte; text:string80); external 'PRTLN.COM';

begin
  ClrScr;
  GotoXY(1,25); write('Line 25');
  GotoXY(1,1);
  Print($8F,'Line 1 - column 1 : cursor --> ');
  read(kbd,ch);
  GotoXY(60,3);
  Print($70,'Line 3 - column 60 : cursor --> ');
  read(kbd,ch);
  GotoXY(1,4);
  PrintLn($81,'Line 4 - column 1 : the cursor was placed at Line 5.');
  read(kbd,ch);
  GotoXY(1,24);
  PrintLn($F0,'Line 24 - column 1 : but it was shifted up to line 23.');
  read(kbd,ch);
end.
```

ส่งท้าย

มีเรื่องเพิ่มเติมที่ผมยังไม่ได้กล่าวถึงกับการใช้กระบวนความทั้งหมดที่เขียนเหล่านี้คือ เครื่องไอบีเอ็มที่จะใช้จะต้องมีโมโนโครมอะแด็ปเตอร์การ์ด หรือโมโน-กราฟิกการ์ด สำหรับคัลเลอร์กราฟิกอะแด็ปเตอร์การ์ดนั้น จะต้องแก้ไขโปรแกรมในเครื่องไอบีเอ็มคอมแพตของผมไม่ได้ใช้ ผมก็เลยไม่กล้าที่จะแก้ไขโปรแกรมเพิ่มเติมหากไม่ได้ทดลองใช้งานจริง ซึ่งหากนำมาเสนอต่อท่านผู้อ่านอาจมีข้อผิดพลาดได้ สุดท้ายหวังว่าท่านผู้อ่านคงจะได้ประโยชน์จากกระบวนความเพิ่มเติมจากบทความนี้ไม่มากก็น้อย สำหรับผมนั้นได้ประโยชน์คุ้มจริงๆ ครับ

หนังสือประกอบบทความ

1) IBM PC/XT TECHNICAL MANUAL REFERENCE
2) TURBO PASCAL VERSION 3.0 MANUAL REFERENCE
3) THE 8086 BOOK

รูปที่ 14 โปรแกรมสำหรับกระบวนความ Print

```
CODE    SEGMENT
        ASSUME  CS:CODE

PRINT   PROC    NEAR
        PUSH    BP
        MOV     BP,SP

        MOV     CH,0
        MOV     CL,BYTE PTR [BP+4]    ; cx = length(text)
        JCXZ    SHORT FINISH          ; if cx = 0 then goto finish
        PUSH    CX                    ; temp1 = cx
        ; read a current cursor position
        MOV     AH,3                  ; ah = 3  (mode 3 = read cursor pos.)
        MOV     BH,0                  ; bh = 0
        INT     10H                   ; interrupt 10H ( result return to DX )
        ; Calculate a location of current cursor position on
        ; video ram
        MOV     AL,80                 ; al = 80 ( 1 line = 80 columns)
        MOV     CL,DH                 ; cl = dh ( DH hold row number )
        IMUL    CL                    ; ax = al*cl
        PUSH    AX                    ; temp2 = ax
        MOV     AL,DL                 ; al = dl ( DL hold col number )
        CBW                           ; ax = al
        POP     BX                    ; bx = temp2
        ADD     AX,BX                 ; ax = ax+bx
        MOV     CL,1                  ;
        SAL     AX,CL                 ; ax = ax*2
        MOV     DI,AX                 ; di = ax ( result of calculation)

        ; point ES to Video ram
        MOV     AX,0B000H  ; ax = 0B000H (starting address of video ram)
        MOV     ES,AX      ; es = ax     ( ES point at 0B000H )
        ; Display on screen
        POP     CX         ; cx = temp1 (number of character to display)
        MOV     SI,BP
AGAIN:  CMP     DI,3840
        JL      SHORT CONT            ; if di < 3840 then goto cont
        PUSH    CX
        PUSH    BP
        CALL    SCROLL
        POP     BP
        POP     CX
        SUB     DI,160
CONT:   MOV     AL,BYTE PTR SS:[SI+5] ; get a character
        MOV     AH,BYTE PTR [BP+85]   ; [BP+85] = Attribute
        MOV     ES:[DI],AX            ; write a character to screen
        INC     DI                    ; di = di+1
        INC     DI                    ; di = di+1
        INC     SI                    ; si = si+1
        LOOPNZ  AGAIN                 ; cx = cx-1 ;if cx<>0 then Goto AGAIN
        ; update cursor position
        MOV     AX,DI                 ; ax = di
        MOV     CL,1
        SAR     AX,CL                 ; ax = ax/2
        MOV     CL,80                 ; cl = 80
        IDIV    CL                    ; ax = ax/cl
        MOV     DH,AL                 ; dh = (ax div cl -> row)
        MOV     DL,AH                 ; dl = (ax mod cl -> column)
        CMP     DH,23
        JLE     SHORT CONT2
        PUSH    DX
        PUSH    BP
        CALL    SCROLL
        POP     BP
        POP     DX
        MOV     DH,23
CONT2:  MOV     AH,2                  ; ah = 2 ( mode 2 -> set cursor pos.)
        MOV     BH,0                  ; bh = 0 ( page 0 )
        INT     10H                   ; interrupt 10H service routine
        JMP     SHORT FINISH

SCROLL  PROC NEAR
        MOV     AH,06H                ; mode 6 = scroll page up
        MOV     AL,01H                ; scroll up 1 line
        MOV     BH,07H
        MOV     CX,0000H              ; upper left corner of scroll
        MOV     DH,17H
        MOV     DL,4FH                ; lower right corner of scroll
        INT     10H
        RET
SCROLL  ENDP

FINISH: MOV     SP,BP
        POP     BP
        RET     83
PRINT   ENDP
CODE    ENDS
        END
```

รูปที่ 15 โปรแกรมสำหรับกระบวนความ **Print Ln**

```
CODE    SEGMENT
        ASSUME  CS:CODE

PRINTLN PROC    NEAR
        PUSH    BP
        MOV     BP,SP

        MOV     CH,0
        MOV     CL,BYTE PTR [BP+4]    ; cx = length(text)
        JCXZ    SHORT FINISH          ; if cx = 0 then goto finish
        PUSH    CX                    ; temp1 = cx
        ; read a current cursor position
        MOV     AH,3                  ; ah = 3  (mode 3 = read cursor pos.)
        MOV     BH,0                  ; bh = 0
        INT     10H                   ; interrupt 10H ( result return to DX
        ; Calculate a location of current cursor position on
        ; video ram
        MOV     AL,80                 ; al = 80 ( 1 line = 80 columns)
        MOV     CL,DH                 ; cl = dh ( DH hold row number )
        IMUL    CL                    ; ax = al*cl
        PUSH    AX                    ; temp2 = ax
        MOV     AL,DL                 ; al = dl ( DL hold col number )
        CBW                           ; ax = al
        POP     BX                    ; bx = temp2
        ADD     AX,BX                 ; ax = ax+bx
        MOV     CL,1                  ;
        SAL     AX,CL                 ; ax = ax*2
        MOV     DI,AX                 ; di = ax ( result of calculation)

        ; point ES to Video ram
        MOV     AX,0B000H  ; ax = 0B000H (starting address of video ram)
        MOV     ES,AX      ; es = ax     ( ES point at 0B000H )
        ; Display on screen
        POP     CX         ; cx = temp1 (number of character to display)
        MOV     SI,BP
AGAIN:  CMP     DI,3840
        JL      SHORT CONT            ; if di < 3840 then goto cont
        PUSH    CX
        PUSH    BP
        CALL    SCROLL
        POP     BP
        POP     CX
        SUB     DI,160
CONT:   MOV     AL,BYTE PTR SS:[SI+5] ; get a character
        MOV     AH,BYTE PTR [BP+85]   ; [BP+85] = Attribute
        MOV     ES:[DI],AX            ; write a character to screen
        INC     DI                    ; di = di+1
        INC     DI                    ; di = di+1
        INC     SI                    ; si = si+1
        LOOPNZ  AGAIN                 ; cx = cx-1 ;if cx<>0 then Goto AGAIN
        ; update cursor position
        MOV     AX,DI                 ; ax = di
        MOV     CL,1
        SAR     AX,CL                 ; ax = ax/2
        MOV     CL,80                 ; cl = 80
        IDIV    CL                    ; ax = ax/cl
        MOV     DH,AL                 ; dh = (ax div cl -> row)
        ADD     DH,1                  ; dh = dh+1
        MOV     DL,0                  ; dl = 0 (column)
        CMP     DH,23
        JLE     SHORT CONT2           ; if dh <= 23 then goto cont2
        MOV     DH,23                 ; dh = 23
        PUSH    DX
        PUSH    BP
        CALL    SCROLL
        POP     BP
        POP     DX
CONT2:  MOV     AH,2                  ; ah = 2 ( mode 2 -> set cursor pos.)
        MOV     BH,0                  ; bh = 0 ( page 0 )
        INT     10H                   ; interrupt 10H service routine
        JMP     SHORT FINISH

SCROLL  PROC NEAR
        MOV     AH,06H                ; mode 6 = scroll page up
        MOV     AL,01H                ; scroll up 1 line
        MOV     BH,07H
        MOV     CX,0000H              ; upper left corner of scroll
        MOV     DH,17H
        MOV     DL,4FH                ; lower right corner of scroll
        INT     10H
        RET
SCROLL  ENDP

FINISH: MOV     SP,BP
        POP     BP
        RET     83
PRINTLN ENDP
CODE    ENDS
        END
```